UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/649,903	08/26/2003	Mahesh A. Ramchandani	5150-77400	5752

7590 03/31/2009

Jeffrey C. Hood
Meyertons, Hood, Kivlin,
Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767

EXAMINER	
MITCHELL, JASON D	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
03/31/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/649,903
Filing Date: August 26, 2003
Appellant(s): RAMCHANDANI, MAHESH A.

Jeffery Hood (#35198)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 12/15/08 appealing from the Office action mailed 9/15/08.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

No amendment after final has been filed.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is substantially correct. The changes are as follows: Claim 99 was rejected in view of Stutz and Carter (see par. 49 of the rejection mailed 9/15/08) and Claim 91 was only intended to be rejected over Grey in view of Stutz and Carter (second grounds to be reviewed; see pars. 53-56 of the rejection mailed 9/15/08). While the claim number was included in the heading of the "Grey in view of Stutz" no rejection of the claim was made in this section.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

6,401,220	Grey et al.	6-2002
5,485,617	Stutz et al.	1-1996
6,718,534	Carter et al.	4-2004

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 76-86, 88-90, 92-107 and 109-114 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,401,220 to Grey et al. (Grey) in view of US 5,485,617 to Stutz et al. (Stutz).

Regarding Claim 76: Grey discloses a computer-implemented method for displaying information regarding a test executive sequence, wherein the test executive sequence includes a plurality of steps, the method comprising:

including a GUI element in a graphical user interface of a run-time operator interface application, wherein the GUI element is operable to display information (Fig. 4 see the 'Main' tab, 'Step' column);

including a control in the run-time operator interface application, wherein the control includes pre-existing first functionality for determining the steps in the test executive sequence (col. 4, lines 4-5 "loading the file, the TestStand Engine automatically determines the type being loaded"; col. 3, lines 23-25 "The TestStand

Engine executes sequences, wherein sequences contain steps"; col. 4, lines 59-62 Instances of the step type incorporate this common functionality … from the step type"); configuring a binding between the GUI element and the control (col. 3, lines 16-18 "The sequence editor … interface[s] to the test executive engine"; 'configuring' here is very broad and encompasses any programming etc. that is done to connect the two objects, thus the sequence editor is 'configured' to interface with the test executive engine), wherein configuring the binding enables the GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application (Fig. 4 see the 'Main' tab, 'Step' column; col. 12, line 65-col. 13, line 6 "In the TestStand sequence editor 212 the user can start multiple concurrent executions. … displays the steps in the currently executing sequence"); and

executing the run-time operator interface application, wherein said executing comprises the control executing to automatically determine the steps in the test executive sequence (col. 4, lines 4-5 "loading the file, the TestStand Engine automatically determines the type being loaded"), wherein the binding between the GUI element and the control causes the GUI element to automatically display at least a subset of the steps in response to the control determining the steps (Fig. 4 see the 'Main' tab, 'Step' column), wherein the GUI element displays the at least a subset of the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application (Fig. 4 see the 'Main' tab, 'Step' column).

Grey does not disclose including the GUI element and control in the run-time operator interface in response to user input.

Stutz teaches that including GUI elements and controls in a run-time operator interface is done in response to user input (col. 10, lines 46-48 "specifies the visual components and their location on the display").

It would have been obvious to one of ordinary skill in the art at the time the invention was made to develop the run-time operator interface disclosed in Grey using the methods taught in Stutz *(col. 10, lines 46-48)* because Stutz provides "an improved method … for dynamically generating object connections" *(col. 8, lines 14-17)*.

Regarding Claim 77: The rejection of claim 76 is incorporated; further Grey discloses:

wherein the control also includes pre-existing functionality for formatting the at least a subset of the steps in the test executive sequence into a formatted list (col. 31, lines 31-33 "specify a name, description, and comment for the step type. The user also can specify an icon and a module adapter"; col. 13, lines 32-33 "The TestStand Test Executive Engine 220 is used for creating … sequences");

wherein the GUI element automatically displaying the at least a subset of the steps comprises the GUI element automatically displaying the list of the at least a subset of the steps (Fig. 4 see the 'Main' tab, 'Step' column).

Regarding Claim 78: The rejection of claim 77 is incorporated; further Grey discloses wherein in performing said formatting the at least a subset of the steps in the test executive sequence into the formatted list, the control is operable to:

determine information regarding each of the at least a subset of the steps in the test executive sequence (col. 31, lines 31-33 "specify a name, description, and comment for the step type"); and

format the information for display in the GUI element (Fig. 4, see 'Main' tab);

wherein the formatted list includes the formatted information for each of the steps in the at least a subset of the steps (Fig. 4, see 'Main' tab).

Regarding Claim 79: The rejection of claim 76 is incorporated; further Grey discloses:

wherein the test executive sequence is stored in a sequence file (col. 5, lines 53-54 "a test sequence file");

wherein in automatically determining the steps in the test executive sequence, the control is operable to automatically obtain information from the sequence file regarding the test executive sequence and determine the steps based on the information obtained from the sequence file (col. 4, lines 1-5 "the user loads a file … The TestStand Engine automatically determines the type being loaded").

Regarding Claim 80: The rejection of claim 76 is incorporated; further Grey discloses:

configuring the control in response to configuration user input after said including the control in the run-time operator interface application (col. 31, lines 31-33 "specify a name, description, and comment for the step type. The user also can specify an icon and a module adapter"), wherein the configuration user input specifies an appearance for the displayed steps, wherein configuring the control enables the control to cause the steps to be displayed in the GUI element with the specified appearance (Fig. 4, see 'Main' tab).

Regarding Claim 81: The rejection of claim 80 is incorporated; further Grey discloses:

wherein the configuration user input specifies one or more properties regarding a plurality of columns to display in the GUI element (col. 31, lines 31-33 "specify a name, description, and comment for the step type. The user also can specify an icon and a module adapter"); wherein configuring the control enables the control to cause information for each displayed step to be displayed in the GUI element in the plurality of columns according to the one or more specified properties (Fig. 4, see 'Main' tab).

Regarding Claim 82: The rejection of claim 76 is incorporated; further Grey discloses:

wherein the GUI element comprises a first GUI element;

wherein the method further comprises:

including a second GUI element in the run-time operator interface application in response to user input (col. 4, lines 4-5 "the user loading the file"; Note that those of ordinary skill in the art would have recognized this action is

preformed through a GUI element; also see col. 23, lines 36-39 “start an execution in the sequence editor by selecting the Run <SequenceName> item”); and

configuring a binding between the second GUI element and the control (col. 3, lines 16-19 “the operator interface programs interface to the test executive engine”);

wherein executing the run-time operator interface application comprises the second GUI element receiving user input during execution of the run-time operator interface application (col. 4, lines 4-5 “the user loading the file”; col. 23, lines 36-39 “start an execution in the sequence editor by selecting the Run <SequenceName> item”);

wherein the binding between the second GUI element and the control causes the control to automatically determine the steps in the test executive sequence in response to the user input received to the second GUI element during execution of the run-time operator interface application (col. 4, lines 4-5 “In response to the user loading the file, the TestStand Engine automatically determines the type being loaded”);

wherein said configuring the binding between the first GUI element and the control enables the first GUI element to automatically display the at least a subset of the steps in response to the user input received to the second GUI element during execution of the run-time operator interface application (Fig. 4 see the ‘Main’ tab, ‘Step’ column).

Regarding Claim 83: The rejection of claim 76 is incorporated; further Grey does not disclose including the control in the run-time operator interface application or that configuring the binding between the GUI elements removes a need for a user to create program code for providing these functionalities.

Stutz teaches including a control in the run-time operator interface application enables a user to configure the run-time operator interface application to perform a first functionality without requiring the user to create program code (col. 10, lines 42-45 "a list of predefined components (objects) that can be interconnected"); and

configuring a binding between a GUI element and the control enables the user to configure the run-time operator interface application to automatically display the results of said first functionality without requiring the user to create program code for displaying the results (col. 11, lines 5-11 "Using the various commands provided by the buttons in the command area 502").

Regarding Claim 84: The rejection of claim 76 is incorporated; further Grey discloses:

wherein the test executive sequence is operable to perform one or more tests on one or more units under test (UUTs) (col. 4, lines 47-48 "A sequence comprises a series of steps, wherein a step is typically a test preformed on an instrument.").

Regarding Claim 85: The rejection of claim 76 is incorporated; further Grey discloses:

wherein the test executive sequence is associated with a test executive environment (col. 13, lines 32-33 “The TestStand Test Executive Engine 220 is used for creating, editing, executing, and debugging sequences.”);

wherein the control is operable to call the test executive environment during execution of the run-time operator interface application to determine the steps in the test executive sequence (col. 3, lines 12-21 “The TestStand Engine interfaces through an adapter interface to … the code modules and sequence files”; col. 13, lines 39-41 “The user can call the Engine API from any programming environment”).

Regarding Claim 86: The rejection of claim 76 is incorporated; further Grey discloses:

wherein the control comprises a software component constructed in accordance with an ActiveX [tm] specification (col. 3, lines 30-33 “The TestStand Engine exports an ActiveX automation API”).

Regarding Claim 88: The rejection of claim 76 is incorporated; further, while not explicitly stated, it is clear from Grey’s disclosure that the control (col. 3, lines 30-33 “The TestStand Engine”) does not appear on the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application.

Regarding Claim 89: The rejection of claim 76 is incorporated; further Grey does not disclose the control is a pre-existing control provided by an application development environment.

Stutz teaches a pre-existing control provided by an application development environment (col. 10, lines 42-45 "a list of predefined components (objects) that can be interconnected").

Regarding Claim 90: The rejection of claim 89 is incorporated; further Grey does not explicitly disclose installing an application development environment on a computer system.

Stutz discloses both the application development environment and the control (col. 10, lines 42-45 "visual programming environment … list of predefined components") "implemented on a computer system" (col. 9, lines 15-21). Accordingly, both the application development environment and the control must have been installed on the computer system

Regarding Claim 92: The rejection of claim 76 is incorporated; further Grey does not disclose the configuring the binding between the GUI element and the control comprises performing one or more calls during execution of the run-time operator interface application.

Stutz discloses said configuring the binding between the GUI element and the control comprises performing one or more calls to bind the GUI element to the control during execution of the test executive application (col. 15, lines 49-52 "The function SetUpConnection … connects the appropriate notification interface").

Regarding Claim 93: The rejection of claim 76 is incorporated; further Grey does not disclose configuring the binding between the GUI element and the control is preformed in response to user input.

Stutz teaches configuring a binding between a GUI element and a control is performed in response to receiving user input to a graphical user interface to specify the binding between the GUI element and the control (col. 11, lines 5-11 "Using the various commands provided by the buttons in the command area 502").

Regarding Claim 94 recites limitations similar to those addressed in the rejection of claim 76 with the exception that the claim is directed to a control and GUI element for respectively generating and displaying a report. (see Grey col. 8, lines 9-10 "The TestStand Engine operates to automatically collect the results of each step in the sequence during execution"; col. 56, lines 48-50 "TestReport … to generate the contents of the test report"; Fig. 50, see the 'Context' tab, 'ResultList' node; also see the 'Report' tab).

Regarding Claim 95 recites limitations similar to those addressed in the rejection of claim 76 with the exception that the claim is directed to a control and GUI element for respectively generating and displaying an execution result. (see Grey col. 8, lines 9-10 "The TestStand Engine operates to automatically collect the results of each step in the sequence during execution"; col. 56, lines 48-50 "TestReport … to generate the contents of the test report"; Fig. 50, see the 'Context' tab, 'ResultList' node; also see the 'Report' tab).

Regarding Claims 96-97, 99-107 and 109-112 recite limitations similar to those addressed in the rejections of claims 76-93 with the exception that the claims are directed to GUI element for receiving user input to cause a control to automatically invoke execution of a test executive sequence. (see Grey col. 23, lines 35-39 "start an execution … by selecting the Run <SequenceName> item").

Regarding Claim 98: The rejection of claim 96 is incorporated; further Stutz discloses:

wherein the control is a first control;

wherein the method further comprises including a second control in the run-time operator interface application in response to user input, wherein the second control includes pre-existing second functionality (col. 11, lines 11-15 "multiple selection list box object 509");

wherein said configuring the binding between the GUI element and the first control also enables the first control to invoke the second control to perform the second functionality (col. 11, lines 11-15 "code for updating the list of files shown in the multiple selection list box object 509").

Regarding Claim 113 recites limitations similar to those addressed in the rejection of claim 96 with the exception that the claim is directed to a GUI element for receiving user input to cause a control to automatically invoke execution of a test executive sequence. (see Grey col. 24, lines 1-4 “The menus … have commands that allow the user to stop execution”).

Regarding Claim 114 recites limitations similar to those addressed in the rejection of claim 76 with the exception that the claim is directed to first and second GUI elements and a control wherein the control opens a dialog box to allow a user to select a test executive sequence in response to user input received to the first GUI element (Fig. 30, see the text box labeled “File Pathname” containing the text “ComputerCPU.seq”), and wherein the second GUI element automatically displays the steps in the test executive sequence (Fig. 4 see the ‘Main’ tab, ‘Step’ column).

Claim 91 is rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,401,220 to Grey et al. (Grey) in view of US 5,485,617 to Stutz et al. (Stutz) and further in view of US 6,718,534 to Carter et al. (Carter).

Regarding Claim 91: The rejection of claim 89 is incorporated; further the Grey-Stutz combination does not disclose installing the control on the computer system after installing the application development environment.

Carter teaches installing a control after installing an application development environment (col. 6, lines 3-5 "The user can import a control from an external source").

It would have been obvious to a person of ordinary skill in the art at the time of the invention to combine the teachings of the Stutz-Grey combination in order to avoid "replication of the control programmability function" and the associated Duplication of work and increased cost. (Carter col. 1, lines 48-52).

(10) Response to Argument

Independent Claim 76

Regarding the limitation:

including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for determining the steps in the test executive sequence;

Starting in the 2nd par. on pg. 11, the appellant states:

> Grey, taken either singly or in combination with Stutz, does not teach these limitations in combination with the other limitations recited in claim 76. With respect to the limitation of, "including a control in the run-time operator interface application in response to user input, wherein the control includes pre-existing first functionality for determining the steps in the test executive sequence," the Examiner states:

> col. 4, lines 4-5 "The TestStand Engine automatically determines the type being loaded"
>
> However, this portion of Grey has nothing to do with a control that includes pre-existing first functionality for determining the steps in the test executive sequence. The cited portion of Grey instead relates to step types. A step type "essentially comprises a custom set of properties and/or operations associated with a step." The user can define various step types. When a given step is created and added to the sequence, the user can request that the step be of a particular step type. Grey teaches that, "For each type that a file uses, the TestStand system stores the definition of the type in the file." (Col. 3, lines 38-40). The teaching cited by the Examiner refers to the file subsequently being loaded after the step type definitions have been stored in it. Grey teaches here that, "In response to the user loading the file, the TestStand Engine automatically determines the type being loaded, and then automatically determines if the loaded type conflicts with one or more previously loaded/registered types." (Col. 4, lines 4-5). Thus, the cited passage teaches nothing whatsoever about a control that includes pre-existing first functionality for determining the steps (not the step type!) in the test executive sequence.

The examiner respectfully disagrees. Grey's identification of the 'type' of step (i.e. identification of the defined functionality of the step see e.g. col. 4, lines 59-64 "The functionality and/or properties defined by a step type") reasonably falls within the scope of the broadly recited "determining the steps in the test executive sequence". In other words, Grey col. 4, lines 4-5 disclosing the intended functionality of a particular step in a sequence of steps (see e.g. col. 4, lines 59-62 "Instances of the step type incorporate this common functionality … from the step type"). The claim only recites "first functionality for determining the steps in the test executive sequence" and does not indicate any particular method of determining the steps or any particular feature by which the steps are determined or the context in which the determining takes place (e.g. while loading or executing a sequence) and thus does not provide any distinction over Gray's determination of a step type (e.g. col. 4, lines 4-5).

Further, regardless of the particular passage of Grey cited in the rejection, the appellant acknowledges that Grey provides a run-time operator interface capable of executing a sequence of steps (see 2nd par. on pg. 10 of the brief). In order to do so Grey must also determine the steps in the sequence. In other words, in order to execute a step, the step (and its associated functionality) must first be 'determined'. This functionality is provided by the "TestStand Engine" (see e.g. col. 3, lines 23-25 "The TestStand Engine executes sequences, wherein sequences contain steps") and thus is provided in a control included in the run-time operator interface GUI.

In the 2nd par. on pg. 12 the appellant states:

> Appellant respectfully disagrees and submits that one skilled in the art would readily recognize the difference between a step and a step type in view of Grey's disclosure. As discussed above, the user can define various step types. When a given step is created and added to the sequence, the user can request that the step be of a particular step type. Thus, the step will automatically be configured with the custom set of properties and/or operations defined for the step type. For example, in FIGs. 16-18 Grey illustrates various tabs of a Step Type Properties dialog box which enables a user to set properties for a Numeric Limit Test step type. After the Numeric Limit Test step type has been defined, the user can then create various instances of Numeric Limit Test steps, i.e., steps of the Numeric Limit Test step type. The steps will automatically inherit the properties that were defined for the Numeric Limit Test step type. Thus, the Numeric Limit Test step type is not itself a step, but is a type which defines properties for steps having that step type.

The examiner respectfully disagrees. First the question is not whether one of ordinary skill in the art would recognize the difference between a step and a step type but whether they would recognize a patentable distinction between *determining* a step and a step type. As the appellant has acknowledged, Grey's 'step types' define the functionality of a particular step instance (see e.g. col. 4, lines 59-64 "The functionality

and/or properties defined by a step type"). Accordingly, by determining the step type for a particular step (e.g. col. 4, lines 51-62 "In a test sequence … Instances of the step type"), Grey is determining the desired functionality for that step. Because the purpose of the claimed step is to indicate a particular test functionality to be preformed, those of ordinary skill in the art would have recognized Grey's determination of a step type indicating the indented functionality for a particular step as patentably indistinct from the claimed limitation.

In the par. bridging pp. 12 and 13 the appellant states:

> Appellant respectfully disagrees. As noted above, Grey teaches, "For each type that a file uses, the TestStand system stores the definition of the type in the file." (Col. 3, lines 38-40). The teaching cited by the Examiner refers to the file subsequently being loaded after the step type definitions have been stored in it. Grey teaches here that, "In response to the user loading the file, the TestStand Engine automatically determines the type being loaded, and then automatically determines if the loaded type conflicts with one or more previously loaded/registered types." Thus, Grey teaches determining the types in the file. As discussed above, step types are defined independently of any particular step, and thus the step type definitions that have been stored in the file can be determined without determining the actual steps in a test executive sequence.

The examiner respectfully disagrees. First it is the examiner's understanding that the file which 'uses' the types in col. 3, lines 38-40 is a file defining a test executive sequence (col. 5, lines 53-57 "The test sequence file ... wherein one or more of the steps are of the first step type"). Further, while step types may be "defined independently of any particular step", at least some steps are instances of a particular step type (col. 5, lines 53-57 "The test sequence file ... wherein one or more of the steps

are of the first step type"). Thus determining the step type for a step instance meets the broadly claimed 'determining the steps'

Regarding the limitation:

configuring a binding between the GUI element and the control, wherein configuring the binding enables the GUI element to automatically display at least a subset of the steps in the test executive sequence in response to the control determining the steps in the test executive sequence during execution of the run-time operator interface application;

Starting in the last partial par. on pg. 13, the appellant states:

> The limitations in question relate to the "at least a subset of the steps" being automatically displayed in the GUI element during execution of the run-time operator interface application. However, the cited portions of Grey teach nothing whatsoever about any steps of the sequence being displayed during execution of the run-time operator interface application. As per Col. 3, lines 16-18, Grey merely teaches here that:
>
> > The sequence editor and the operator interface programs interface to the test executive engine, referred to as the TestStand Engine. One or more process models couple to the TestStand Engine. The TestStand Engine interfaces through an adapter interface to one or more adapters. The adapters in turn interface to code modules and sequence files.
>
> Thus, Grey is here simply describing the architecture of the test executive system, teaching that the sequence editor and the operator interface programs interface to the test executive engine, e.g., as shown in FIG. 2. In contrast, claim 76 recites, "configuring a binding between the GUI element and the control". The cited portions of Grey teach nothing about configuring a binding between a GUI element and a control.

The examiner respectfully disagrees. Again the limitation is broad. The claim only recites “configuring a binding” and the resultant functionality. It does not describe what configuration steps are preformed or how the binding is implemented. Grey col. 3, lines 16-18 disclose that the architecture of the test executive system includes a binding configured between a GUI element (the sequence editor and particularly the ‘main’ tab

disclosed in Fig. 4) and a control (the TestStand Engine). In other words by defining the interface and the interactions across the interface a developer has 'configured a binding' of the two objects. Further, this 'binding' enables the functionality (i.e. displaying the steps as shown in fig. 4) of the sequence editor (col. 13, lines 32-33 "The TestStand Test Executive Engine 220 is used for creating, editing, executing, and debugging sequences"). Additionally, those of ordinary skill would have recognized that the functionality defined by the code is only provided when the code is executed (i.e. during execution of the run-time operator interface).

In the second full par. on pg. 14, the appellant states:

> As per the "Main" tab and the "Step" column in FIG. 4 cited by the Examiner, this is a screen shot illustrating Grey's sequence editor. More particularly, the screen shot illustrates the steps in the default process model. (See Col. 21, lines 48-50). A process model is basically a series of steps that are commonly used by different test sequences and acts as a template so that the user does not have to create the same steps every time for different sequences. Grey teaches that, "A process model is in the form of a sequence file. The user can edit a process model just as he/she edits other sequences." (Col. 20, lines 20-22). Thus, the screen shot in FIG. 4 illustrates the sequence editor and relates generally to creating the sequence. In contrast, the recited limitations in claim 76 relate to the "at least a subset of the steps" being automatically displayed during execution of the run-time operator interface application in response to the control determining the steps, where the binding configured between the GUI element and the control enable the GUI element to automatically display the "at least a subset of the steps". Grey simply does not teach this subject matter, taken either singly or in combination with Stutz.

The examiner respectfully disagrees. Grey's sequence editor provides all of the functionality claimed for an operator interface (e.g. determining and displaying the steps of a sequence), the sequence editor anticipates the claimed 'operator interface' regardless of the particular term used to describe it. Further, the 'main' tab of figure 4

displays “at least a subset of the steps” (see e.g. col. 21, lines 48-50 “Fig. 4 shows the entire set of steps … in the default process model”); and the steps are clearly displayed during execution of the run-time operator interface application (an application does not provide any functionality until it is executed). Thus Grey discloses a run-time operator interface displaying the steps as claimed.

To the extent that the appellant may argue the sequence editor is patentably distinct from the claimed "run-time operator interface application", the examiner points to Grey col. 12, line 65-col. 13, line 6:

> In the TestStand sequence editor 212 the user can start multiple concurrent executions. In trace mode, the execution window displays the steps in the currently executing sequence.

and Col. 13, lines 16-25:

> Like the sequence editor 212, the run-time operator interfaces 202 allow the user to start multiple concurrent executions, set breakpoints, and single step

From these citations it should be clear that both the “sequence editor 212” and the “run-time operator interfaces 202” display at least a subset of the steps during execution of their comprising applications. Accordingly, either of Gray’s “sequence editor” or “run-time operator interface” meets the claim limitation.

In the first par. on pg. 15, the appellant states:

> Appellant respectfully disagrees. FIG. 4 does not show an operator interface application, as asserted by the Examiner. FIG. 4 instead illustrates Grey's sequence editor, as noted above. As discussed at the beginning of the arguments with respect to claim 76, Grey teaches that the sequence editor is "a program that provides a graphical user interface for creating, editing, and debugging sequences." (Col. 2, lines 3-5). More specifically, the sequence editor allows the user to create a sequence of steps, e.g., where different steps in the sequence call different test

> modules to perform specific tests on the unit under test. (Col. 1, lines 52-64). A run-time operator interface program is "a program that provides a graphical user interface for executing sequences on a production station." (Col. 2, lines 6-8). Thus, a user first creates a test sequence using the sequence editor. The sequence is then executed in a run-time operator interface program.

The examiner respectfully disagrees. First it is noted that, at least claim 76, does not recite “executing sequences on a production station". Regardless, Grey discloses “In the TestStand sequence editor 212 the user can start multiple concurrent executions” (col. 12, line 65-col. 13, line 6; also see col. 13, lines 16-18 "the user can use the TestStand sequence editor 212 at a production station"). Accordingly it should be seen that Grey's ‘sequence editor' is capable of executing sequences in addition to creating, editing, and debugging them. As shown in the rejection and discussed above, Grey's 'sequence editor' provides all of the functionality of the claimed 'run-time operator interface' and thus differs only in the lexicon used to describe it. Accordingly there is no patentable distinction.

Regarding the limitation:

> *executing the run-time operator interface application, wherein said executing comprises the control executing to automatically determine the steps in the test executive sequence, wherein the binding between the GUI element and the control causes the GUI element to automatically display at least a subset of the steps in response to the control determining the steps, wherein the GUI element displays the at least a subset of the steps in the graphical user interface of the run-time operator interface application during execution of the run-time operator interface application.*

The appellant’s arguments regarding this limitation refer to and re-apply the arguments discussed above (see the last partial par. on pg. 15 – second full par. on pg. 16), and are addressed similarly.

Regarding the combination of Grey and Stutz:

It would have been obvious to one of ordinary skill in the art at the time the invention was made to develop the run-time operator interface disclosed in Grey using the methods taught in Stutz (col. 10, lines 46-48) because Stutz provides "an improved method ... for dynamically generating object connections (col. 8, lines 14-17).

In the last par. on pg .16, the appellant states:

> Stutz relates generally to creating connections between objects in a program. Stutz's invention operates at a fairly low level of programming, e.g., in order to allow a source object to notify a sink object. (See Abstract). In contrast, Grey's invention operates at a much higher level and relates to a test executive system that allows a user to create a test sequence, e.g., through the graphical user interface of a sequence editor such as illustrated in FIG. 4. It is difficult to see the particular relevance of Stutz's invention to Grey's invention or to understand how or why Stutz's teaching of dynamically generating object connections would be incorporated into Grey's system.

The examiner respectfully disagrees. First it should be noted that Stutz discloses a user including, configuring and binding GUI elements and controls (see e.g. col. 10, lines 46-51 “a visual programmer specifies the visual components … the interconnections between various ports”). It is this user action aspect that is missing from the Grey reference (at least in regard to claim 76). In other words Grey discloses a GUI and control already included, configured and bound in an application. Those of ordinary skill in the art would have understood that the application disclosed in Grey must at some time have been developed before it could provide the disclosed functionality. Stutz discloses a method of developing an application which would have been recognized as applicable and advantageous to the development of Grey’s application (see .e.g. Stutz col. 8, lines 14-17 “an improved method ... for dynamically generating object connections").

Regarding the appellant's general assertions against the rejection of claim 76 and the other independent claims.

In the par. bridging pp. 17 & 18 the appellant states:

> Grey teaches a test executive system which includes a built-in sequence editor and default run-time operator interfaces. The built-in sequence editor is operable to display the steps in a test sequence, e.g., as shown in FIG. 4. However, Grey does not contain any teaching regarding the user of the test executive system creating his own run-time operator interface application by including in the run-time operator interface application a control with pre-existing functionality to automatically determine and display the steps in the sequence in a GUI element, as recited in claim 76. Furthermore, Stutz does not remedy this deficiency of Grey's teaching.

The examiner respectfully disagrees. The claims do not recite “the user of the test executive system creating his own run-time operator interface application”. Further, the claims do not support the implied argument that the inclusion of the GUI element and control is preformed by and/or during the execution of a test executive application. Instead the claims only require the development of operator interface (“including a GUI element … including a control … configuring a binding … and executing the run-time operator interface application”). Further, when we look to the appellant's specification we see that these development steps are disclosed as taking place in a standard development environment. See for example pg. 27, lines 9-27

> including the GUI element(s) … in response to user input received to the graphical user interface of the ADE … including the control(s) ... in response to user input received to the graphical user interface of the ADE

and pg. 2, lines 21-23

> Application Development Environment (ADE)--A programming environment such as LabVIEW, LabWindows/CVI, Microsoft Visual C++, Microsoft Visual Basic, etc., in which the user can create test modules and run-time operator interfaces.

The proposed combination of Grey and Stutz meets these limitations as indicated in the rejection and discussed above.

Dependent Claims 77 and 78

In the last par. on pg. 19 the appellant states:

> This pertains to the user defining a step type such as described above and teaches nothing whatsoever about a control that can be included in a run-time operator interface application, where the control includes pre-existing functionality for formatting the at least a subset of the steps into a formatted list.

The examiner respectfully disagrees. Grey col. 31, lines 31-33 describe defining the way the step type should be displayed (i.e. its format) in the 'Main' tab of Fig. 4 (e.g. Name and Icon in the first column, Description in the second etc.). Further, as discussed in above the TestStand Engine (the control) provides the described functionality (col. 13, lines 32-33 "The TestStand Test Executive Engine 220 is used for creating … sequences").

Dependent Claims 80 and 81

Starting in the 4th par. on pg. 20 the appellant states:

> As discussed above, this pertains to the user defining a name, description, and comment for a step type. Grey teaches nothing whatsoever about configuring a control in response to configuration user input that specifies an appearance for the steps that are displayed in the GUI element, where configuring the control enables the control to cause the steps to be displayed in the GUI element with the specified appearance.
>
> The Examiner cites the same teaching in the rejection of claim 81. However, Grey teaches nothing regarding the specific limitations of, "wherein the configuration user input specifies one or more properties regarding a plurality of columns to display in the GUI element" and "wherein configuring the control enables the control

> to cause information for each displayed step to be displayed in the GUI element in the plurality of columns according to the one or more specified properties."

The examiner respectfully disagrees. As discussed above Grey col. 31, lines 31-33 discloses specifying how a step type (and thus step instances) will be displayed in Fig. 4. Further, this is done by specifying one or more properties regarding a plurality of columns (e.g. Name and Icon in the first column, Description in the second etc.).

Dependent Claim 82

In the 2nd to last par. on pg. 21, the appellant states:

> In the rejection of claim 82 the Examiner cites the same portions of Grey already discussed above. However, Grey does not teach the specific combination of limitations recited in claim 82, in combination with the other recited limitations of claim 82. In particular, there is no teaching regarding the bindings configured between the GUI elements and the control and the recited functionality which is caused by the bindings.

The examiner respectfully disagrees. Claim 82 repeats much of the functionality of the independent claims (e.g. configuring a binding between a [second] GUI element and a control). Accordingly it is appropriate for the rejection to be based on many of the same disclosures. Further, as noted in the rejection, the claimed first and second GUI elements bound to a control are addressed by a user selectable menu item (second GUI element) which causes the TestStand engine (control) to load and/or execute a sequence in the sequence editor (first GUI element). This arrangement is disclosed by Grey (see e.g. Col. 23, 36-39 “The user can start an execution in the sequence editor by selecting the Run <SequenceName> item").

Dependent Claim 85

In the 2nd to last par. on pg. 22, the appellant states:

> Thus, the Examiner has referred to Grey's test executive engine in both cases. However, claim 85 recites that a control calls the test executive environment. The Examiner has equated the test executive environment with the engine itself. What then does the Examiner consider to be the control? It is not at all clear. In any case, Grey does not teach a control which calls the test executive engine (which the Examiner has equated with the recited "test executive environment") and where the control includes pre-existing first functionality for determining the steps in the test executive sequence, as recited in claims 85 and 76.

The examiner respectfully disagrees. Initially it is noted that the term "environment" is exceptionally broad and it should be clear that any system which executes a test can reasonably be read on the claimed "test executive environment". Accordingly, those of ordinary skill in the art would have recognized that Grey's system in its entirety represent a test executive environment. Further, Grey discloses "The TestStand Engine interfaces through an adapter interface to one or more adapters The adapters in turn interface to the code modules and sequence files" (col. 3, lines 12-21). Those of ordinary skill in the art would have recognized this interface between the control ("TestStand Engine") and the rest of the application (e.g. "code modules and sequence files") meets the claimed limitation.

Dependent Claim 91

In the second to last par. on pg. 9 the appellant states:

> Dependent claim 91 is indicated as being rejected in the listing of claims in the Office Action. However, the Examiner's remarks do not address claim 91, and no rationale is given for its rejection. Appellant thus submits that claim 91 does not stand properly rejected.

The examiner agrees, but notes that claim 91 is rejected over Grey in view of Stutz further in view of Carter (see e.g. pg. 26 of the action mailed 9/15/08).

Independent Claims 94, 95, 96, 97 and 113 (see pp. 23-27)

The appellant's arguments regarding these claims refer to and re-apply the arguments against claim 76 (particularly a user of the test executive application including the GUI element and control) which are discussed above, and are addressed similarly.

Dependent Claim 98

In the 3rd par. on pg. 27 the appellant states:

> The Examiner gives no rationale for rejecting claim 98 and does not address the specific recited limitation of, "wherein said configuring the binding between the GUI element and the first control also enables the first control to invoke the second control to perform the second functionality." Appellant submits that the cited references do not teach this limitation in combination with the other recited limitations.

The examiner respectfully disagrees. This omission was corrected in the action mailed 9/15/09. Specifically the limitation in question is addressed with Stutz col. 11, lines 11-15 "code for updating the list of files shown in the multiple selection list box object 509".

Dependent Claim 102

The appellant's arguments regarding this claim refer to and re-apply the arguments against claim 85 which are discussed above, and are addressed similarly.

Independent Claim 114

The appellant's arguments regarding this claim refer to and re-apply the arguments against claim 76 (particularly a user of the test executive application including the GUI element and control) which are discussed above, and are addressed similarly.

Section 103 Rejection of claim 91 (Grey in view of Stutz and Carter)

The appellant's arguments regarding these claims refer to and re-apply the arguments against parent claims 89 and 76 which are discussed above, and are addressed similarly.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/Jason Mitchell/
Examiner, Art Unit 2193

Conferees:

/Lewis A. Bullock, Jr./
Supervisory Patent Examiner, Art Unit 2193

/Eddie C. Lee/
Supervisory Patent Examiner, TC 2100